ซอฟต์แวร์

เป็นศาสตร์ที่เกี่ยวข้องกับการใช้กระบวนการทางวิศวกรรมในการดูแลการผลิตซอฟต์แวร์

ซอฟต์แวร์มีบทบาทสำคัญ

📌คือผลิตภัณฑ์

ซอฟต์แวร์เป็นตัวประมวลผลข้อมูลข่าวสารร่วมกับฮาร์ดแวร์คอมพิวเตอร์และสิ่งแวดล้อมต่าง ๆ ให้ดำเนินการ จัดการข้อมูล ปรับเปลี่ยน แสดงผล หรือ การส่งต่อข้อมูล เพื่อสนองต่อความต้องการของผู้ใช้

📌เครื่องมือที่ก่อให้เกิดผลิตภัณฑ์

ซอฟต์แวร์มีบทบาทเป็นตัวควบคุมคอมพิวเตอร์ ตัวสื่อสารข้อมูลสารสนเทศ เป็นเครื่องมือที่ช่วยสร้างเทคโนโลยีใหม่

การนิยาม "ซอฟต์แวร์"

ซอฟต์แวร์ คือ ชุดคำสั่งหรือโปรแกรมคอมพิวเตอร์ที่ใช้สั่งงานให้คอมพิวเตอร์ทำงานตามลำดับขั้นตอน

ซอฟต์แวร์ คือ โครงสร้างข้อมูลซึ่งทำให้โปรแกรมสามารถจัดการสารสนเทศได้

ซอฟต์แวร์ คือ ชุดเอกสารที่บรรยายการปฏิบัติและการใช้โปรแกรม

การนิยามซอฟต์แวร์ในมุมมองวิศวกรรมซอฟต์แวร์คือซอฟต์แวร์ที่ผ่านกระบวนการที่เป็นที่ยอมรับรวมถึงการตรวจสอบความถูกต้องของการประกันคุณภาพเพื่อให้ได้ซอฟต์แวร์ที่มีคุณภาพเป็นไปตามความต้องการของผู้ใช้

คุณลักษณะของซอฟต์แวร์ที่มีคุณภาพ

1. ความถูกต้อง ความสามารถของซอฟต์แวร์ที่ประมวลผลตามหน้าที่ได้ออกแบบไว้อย่างถูกต้อง และเป็นไปตามข้อกำหนดซอฟต์แวร์ที่ระบุไว้
2. ประสิทธิภาพ เป็นการใช้งบประมาณหรือทรัพยากรสิ้นเปลืองน้อยที่สุด ได้แก่ การใช้หน่วยความจำพื้นที่การจัดเก็บข้อมูล
3. ความน่าเชื่อถือ เป็นความสามารถของซอฟต์แวร์ในการทำงานตามฟังก์ชันได้ ภายใต้เงื่อนไข
4. ความสามารถในการใช้งาน ซึ่งต้องคำนึงถึงหลักการออกแบบส่วนการติดต่อผู้ใช้ UI
5. ความง่ายต่อการปรับตัวได้ ความสามารถของการนำซอฟต์แวร์ไปใช้ในสิ่งแวดล้อมอื่นโดยไม่ต้องดัดแปลงหรือแก้ไขซอฟต์แวร์
6. ความสามารถนำกลับมาใช้งานใหม่ได้ ในการนำบางส่วนของซอฟต์แวร์ไปใช้ในระบบซอฟต์แวร์อื่นได้
7. ความเข้ากันได้กับระบบที่แตกต่าง สามารถนำระบบไปใช้ งานกับระบบอื่น ๆ ได้
8. ความสะดวกในการเคลื่อนย้าย ง่ายในการดัดแปลงแก้ไขระบบ เพื่อให้ทำงานได้ในสภาวะแวดล้อมต่างจากที่ได้รับการออกแบบมาโดยเฉพาะ
9. ความปลอดภัย การป้องกันการเข้าถึงข้อมูลต่าง ๆที่ ประมวลผลอยู่ในระบบซอฟต์แวร์

การจัดประเภทซอฟต์แวร์

ซอฟต์แวร์ระบบ

หมายถึงโปรแกรมที่มีหน้าที่ติดต่อการทำงานระหว่างฮาร์ดแวร์กับซอฟต์แวร์ประยุกต์

- ระบบปฏิบัติการ
  มีหน้าที่ควบคุมการปฏิบัติงานของซอฟต์แวร์ เช่น Linux ,windows
- ยูทีลิตี้
  มีหน้าที่ ทำหน้าที่เพิ่มประสิทธิภาพของเครื่องคอมทำให้เครื่องทำงานง่ายขึ้นป้องกันการรบกวนเช่นโปรแกรมป้องกันไวรัส
- ดีไวซ์ไดเวอร์
  มีหน้าที่ติดต่อกับคอมพิวเตอร์ในส่วนการรับเข้าและส่งออกของแต่ละอุปกรณ์
- ตัวแปลภาษา
  มีหน้าที่ แปลภาษาระดับต่ำหรือระดับสูงเพื่อให้คอมพิวเตอร์เข้าใจ

ซอฟต์แวร์ประยุกต์

หมายถึงโปรแกรมที่ใช้สำหรับทำงานต่างๆ

- ซอฟต์แวร์สำหรับงานเฉพาะด้าน
  เป็นซอฟต์แวร์ที่ใช้เฉพาะด้านเช่นซอฟต์แวร์สำหรับงานธนาคาร
  ซอฟต์แวร์สำหรับงานทะเบียน
- ซอฟต์แวร์สำหรับงานทั่วไป
  โดยในซอฟต์แวร์ 1 ตัวมีความสามารถในการทำงานได้หลายอย่าง เช่น ซอฟต์แวร์งานด้านเอกสาร (Microsoft Word)

ซอฟต์แวร์แบบฝัง

เป็นซอฟต์แวร์ที่ฝังติดตั้งไว้ในอุปกรณ์อิเล็กทรอนิกส์ต่างๆ เพื่อควบคุมการทำงานของอุปกรณ์ โดยมีไมโครโพรเซสเซอร์ หรือไมโครคอนโทลเลอร์เป็นหัวใจหลักในการประมวลผลการทำงาน มักพบอยู่ในรูปของส่วนควบคุมการทำงานของอุปกรณ์ต่างๆ เครื่องมือวัดทางการแพทย์ โทรศัพท์มือถือ เป็นต้น

ซอฟต์แวร์ด้านวิทยาศาสตร์และวิศวกรรม

เช่น ดาราศาสตร์ ชีววิทยา ฟิสิกส์ เคมี และงานเชิงตัวเลขต่าง ๆ

ซอฟต์แวร์สายการผลิต

เป็นซอฟต์แวร์เฉพาะด้านที่ลูกค้าหลากหลายประเภท สามารถใช้งานได้ร่วมกัน เช่น โปรแกรมเพื่อการควบคุมสินค้าคงคลัง ซอฟต์แวร์การจัดการฐานข้อมูล และซอฟต์แวร์ควบคุมการเงินต่าง ๆ

ซอฟต์แวร์ปัญญาประดิษฐ์

เป็นซอฟต์แวร์ที่ถูกออกแบบกระบวนการเรียนรู้ของอัลกอริทึมให้ทำงานเลียนแบบสมองของมนุษย์ เพื่อการแก้ปัญหาที่ซับซ้อนสูงตามหลักการและเหตุผล เช่น หุ่นยนต์

เว็บแอปพลิเคชัน

เป็นซอฟต์แวร์ที่ใช้สำหรับการทำงานโต้ตอบผ่านเว็บไซต์ มีการดำเนินการด้านการจัดการฐานข้อมูล โดยไม่จำเป็นต้องมีการติดตั้งโปรแกรมทางฝั่งผู้ใช้ เช่น เว็บเมลล์ การประมูลออนไลน์ โปรแกรมการสนทนา เป็นต้น

today
MOOD

วิศวกรรมซอฟต์แวร์

วิวัฒนาการของวิศวกรรมซอฟต์แวร์

ช่วงที่ 1 (พ.ศ.2488-2508)
มีการใช้จริงในช่วง พ.ศ.2493-2503 เพื่อการผลิตซอฟต์แวร์
NATO (North Atlantic Treaty Organization) จัดประชุม “The NATO Software Engineering Conference” ขึ้น 2 ครั้ง
ครั้งแรกจัดใน ค.ศ.1968 (พ.ศ.2511) ณ ประเทศเยอรมัน และครั้งต่อมา จัดในปี ค.ศ.1969 (พ.ศ.2512) ณ ประเทศอิตาลี

ช่วงที่ 2 (พ.ศ.2508-2528)

เป็นช่วงวิกฤติซอฟต์แวร์ หรือที่เรียกกันว่า วิกฤติซอฟต์แวร์ (Software Crisis) มีผลมาจากการพัฒนาซอฟต์แวร์ต่าง ๆ เกิดปัญหาขึ้นในหลาย ๆ โครงการในช่วงปี ค.ศ.1960-1980 เนื่องจากการจัดการต้นทุนของการทำโครงการเกินงบประมาณ รวมถึงความไม่เหมาะสมของระยะเวลาในการดำเนินงาน เป็นผลให้การดำเนินการพัฒนาซอฟต์แวร์ล้มเหลว

วิกฤติซอฟต์แวร์ (Software Crisis)

- ซอฟต์แวร์ระบบปฏิบัติการ OS/360 ใช้เวลาในการพัฒนาถึง 10 ปี (ค.ศ.1960-1970) โดยมีการลงทุนงบประมาณถึงพันล้านเหรียญ และใช้จำนวนวิศวกรซอฟต์แวร์ถึง 1,000 คน
- ซอฟต์แวร์ระบบรักษาความปลอดภัยของฐานจรวดนำวิถีแอร์เรียน เกิดช่องโหว่ทำให้อาชญากรสามารถขโมยรหัสผ่านในการควบคุมฐานปล่อยจรวดไปได้ ซึ่งต่อมาทำให้เกิดการระเบิดในเวลาต่อมา
- ซอฟต์แวร์ควบคุมระบบการแผ่รังสีสำหรับเครื่องรักษาผู้ป่วยด้วยรังสีวิทยา ระบบผิดพลาดทำให้กัมมันตรังสีจากหลอดบรรจุกัมมันตรังสีเกิดการรั่วไหล แผ่เข้าสู่ร่างกายผู้ป่วยและสุดท้ายทำให้ผู้ป่วยเสียชีวิต

ช่วงที่ 3 (พ.ศ.2528-ปัจจุบัน) ช่วงฟองสบู่แตก

นำเทคโนโลยีสมัยใหม่และวิธีการพัฒนาซอฟต์แวร์ในรูปแบบต่าง โดยใช้

- เครื่องมือ (Tools) สำหรับอำนวยความสะดวกในการพัฒนาซอฟต์แวร์ที่มีการคิดค้นและพัฒนาขึ้น
- สหวิทยาการ (Discipline) ที่เป็นการรวมศาสตร์และศิลป์เข้าด้วยกัน
- วิธีการที่ถูกแบบแผน (Formal Method) ที่เป็นที่ยอมรับด้านวิศวกรรมที่เหมาะสมมาใช้ในกระบวนการผลิต (Process)
- แนวทางในการปรับปรุงกระบวนการผลิต
- ความเป็นมืออาชีพ (Professionalism) โดยเน้นการพัฒนาซอฟต์แวร์ที่ถูกจริยธรรม โดยวิศวกรซอฟต์แวร์จะต้องคำนึงถึงจรรยาบรรณวิชาชีพ

วิศวกรรมซอฟต์แวร์

ความหมายของวิศวกรรมซอฟต์แวร์

นิยามโดย IEEE 610.2

“วิศวกรรมซอฟต์แวร์ หมายถึง กระบวนการที่เป็นระบบ มีหลักการ มีคุณภาพ เพื่อการพัฒนา ดำเนินการ และบำรุงรักษาซอฟต์แวร์ โดยใช้กระบวนการทางวิศวกรรมศาสตร์ในการพัฒนาซอฟต์แวร์ดังกล่าว” (IEEE 610.2)

1. กระบวนการในการออกแบบ พัฒนา และทดสอบซอฟต์แวร์เพื่อให้เป็นไปตามต้องการของลูกค้าหรือผู้ใช้
2. ซอฟต์แวร์ที่ผ่านกระบวนการทางวิศวกรรม ซึ่งทำให้ได้ผลิตภัณฑ์ที่มีการออกแบบอย่างเหมาะสม ผ่านการทดสอบตามแนวทางทางวิศวกรรม
3. มีคุณภาพและความปลอดภัย
4. มีการใช้ศาสตร์ทางด้านคณิตศาสตร์มาช่วยในการออกแบบและตรวจสอบกระบวนการของผลิตภัณฑ์ซอฟต์แวร์ ซึ่งระดับความลึกของศาสตร์ทางด้านคณิตศาสตร์จะขึ้นอยู่กับชนิดของซอฟต์แวร์ว่ามีความซับซ้อน หรือเป็นซอฟต์แวร์ที่ต้องการความปลอดภัยสูงเป็นลักษณะของซอฟต์แวร์วิกฤต (Security critical software)
5. มีการนำหลักการของการจัดการโครงการซอฟต์แวร์ (Software Project Management)นำมาใช้ในการพัฒนาซอฟต์แวร์
6. มีการนำหลักการของการสนับสนนและบำรงรักษาซอฟต์แวร์มาใช้

ความสำคัญของวิศวกรรมซอฟต์แวร์

1. กระบวนการผลิตซอฟต์แวร์สามารถทำได้รวดเร็วขึ้น จะทำให้ระยะเวลาในการผลิต รวมถึงต้นทุนด้านแรงงานคนลดลง ผลิตภัณฑ์เสร็จเร็วขึ้น
2. การเปลี่ยนแปลงในยุคอุตสาหกรรมผลิตอุปกรณ์ทางด้านฮาร์ดแวร์คอมพิวเตอร์มีมากขึ้น ทำให้อุปกรณ์ถูกลง ดังนั้นจึงมีการพัฒนาซอฟต์แวร์ต่าง ๆ เพื่อนำไปใช้งานได้เพิ่มมากขึ้น ทำให้ต้นทุนการบำรุงรักษาซอฟต์แวร์เพิ่มมากขึ้นด้วย
3. วิวัฒนาการของระบบการสื่อสาร งานด้านเครือข่ายมีการพัฒนามากยิ่งขึ้น
4. วิธีการพัฒนาและออกแบบซอฟต์แวร์ เช่น การออกแบบเชิงวัตถุ
5. การเปลี่ยนแปลงการพัฒนาส่วนการติดต่อกับผู้ใช้ในรูปแบบกราฟิกมีมากขึ้น และเน้นการทำงานแบบโต้ตอบได้มากขึ้น
6. แบบจำลองในรูปแบบเดิม ๆ เช่น แบบจำลองเชิงน้ำตก ไม่สามารถคาดการณ์การยอมรับซอฟต์แวร์ของผู้ใช้ได้

วิศวกรรมซอฟต์แวร์ ได้กลาย เป็นศาสตร์และแขนงหนึ่งของการศึกษาเฉพาะ ในการสร้างซอฟต์แวร์ที่มีคุณภาพ เป็นไปตามความต้องการของผู้ใช้ โดย

- ราคาถูกลง
- เป็นที่ยอมรับได้
- ดูแลรักษาได้ง่าย
- มีการประยุกต์ความรู้และเทคโนโลยีทางด้านวิศวกรรมศาสตร์ วิศวกรรมคอมพิวเตอร์ วิศวกรรมระบบ วิทยาการคอมพิวเตอร์ เทคโนโลยีสารสนเทศ การบริหารจัดการโครงการ และความรู้ในขอบเขตที่เกี่ยวข้อง เพื่อสร้างซอฟต์แวร์ได้ตามเป้าหมายภายใต้เงื่อนไขที่กำหนด

วิศวกรรมซอฟต์แวร์

องค์ประกอบของวิศวกรรมซอฟต์แวร์

การวิศวกรรมการผลิต

กระบวนการรับความต้องการของระบบใด ๆ มาพัฒนาเป็นซอฟต์แวร์ โดยมีกระบวนการเป็นลำดับขั้นตอนแสดงดังนี้

1. วิเคราะห์ความต้องการและสร้างข้อกำหนดของระบบ (Requirement Analysis and Specification)
2. จัดทำข้อกำหนดความต้องการซอฟต์แวร์ (Software Requirement Specification)
3. ออกแบบสถาปัตยกรรมซอฟต์แวร์ (Software Architecture Design)
4. พัฒนาซอฟต์แวร์ (Software Development)
5. ทดสอบซอฟต์แวร์ (Software Testing)
6. นำซอฟต์แวร์ไปใช้งาน (Deployment)
7. บำรุงรักษาซอฟต์แวร์ (Software Maintenance)

การวิศวกรรมระบบ กระบวนการศึกษาและวิเคราะห์ระบบที่มีความซับซ้อน สำหรับสนับสนุนการทำงานของวิศวกรรมการผลิต ซึ่งมีลำดับของกระบวนการดังนี้

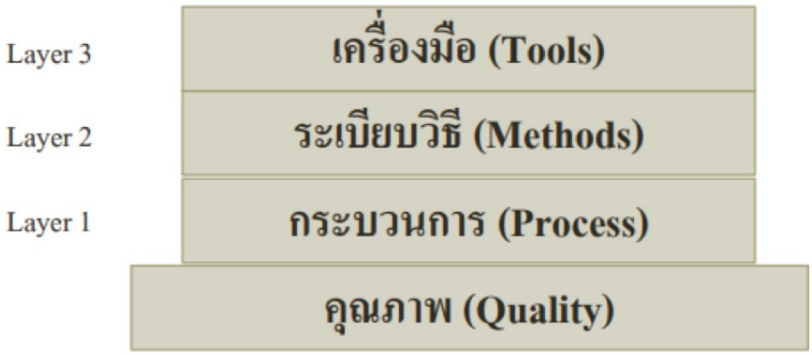
1. กำหนดวัตถุประสงค์ของระบบ
2. กำหนดขอบเขตของระบบ
3. วางแผนแบ่งระบบออกตามฟังก์ชันของระบบ
4. วิเคราะห์ความสัมพันธ์ของส่วนประกอบต่างๆ
5. กำหนดความสัมพันธ์ของข้อมูลเข้า การประมวลผล และผลลัพธ์
6. กำหนดเครื่องมือ ทั้งทางฮาร์ดแวร์ ซอฟต์แวร์
7. กำหนดความต้องการในเชิงการปฏิบัติงานหรือการใช้งานระบบ
8. สร้างแบบจำลองเพื่อนำไปพัฒนาระบบงาน
9. แลกเปลี่ยนข้อคิดเห็นร่วมกันในผู้ที่เกี่ยวข้องกับระบบ

บุคคลด้านวิศวกรรมซอฟต์แวร์

1. **ผู้ใช้ (User)** ซึ่งเป็นผู้ใช้ระบบงานที่แท้จริงที่ต้องใช้งานซอฟต์แวร์ที่ผลิตขึ้นมาเป็นงานประจำ
2. **ลูกค้า (Customer)** อาจเป็นบุคคล องค์กร หรือบริษัท ที่ต้องการให้มีการผลิตซอฟต์แวร์และเป็นผู้จ่ายเงิน
3. **นักพัฒนา (Developer)** อาจเป็น บุคคล บริษัท หรือองค์กรก็ได้เช่นกัน ที่ทำหน้าที่สร้างซอฟต์แวร์ ซึ่งผู้ที่มีหน้าที่หลักควบคุมดูแลนักพัฒนาให้ทำงานตามกระบวนการวิศวกรรมซอฟต์แวร์ก็คือ **วิศวกรซอฟต์แวร์**

กระบวนการซอฟต์แวร์

กระบวนการซอฟต์แวร์ หรือ หมายถึง ลำดับขั้นตอนการผลิตซอฟต์แวร์ที่มีคุณภาพ



กระบวนการซอฟต์แวร์ มักจะถูกนำเสนอในรูปแบบของแบบจำลองกระบวนการ

- เพื่อให้สามารถอธิบายหลักการเฉพาะในแต่ละรูปแบบ
- ผู้พัฒนาสามารถยึดถือเป็นแนวทางในการร่วมพัฒนาซอฟต์แวร์ที่เข้าใจได้ตรงกัน
- อยู่ภายใต้ข้อจำกัด เวลาในการพัฒนา ต้นทุน ขนาดของซอฟต์แวร์ และความต้องการของผู้ใช้งานนั้นเป็นหลัก

กิจกรรมพื้นฐาน

- การสร้างข้อกำหนดซอฟต์แวร์ เป็นการระบุหน้าที่ต่าง ๆ ของซอฟต์แวร์ รวมถึงข้อจำกัดต่าง ๆ
- การออกแบบและการพัฒนซอฟต์แวร์ เป็นการออกแบบสถาปัตยกรรมของซอฟต์แวร์ ส่วนการติดต่อผู้ใช้เพื่อนำไปพัฒนาเป็นซอฟต์แวร์ตามข้อกำหนด
- การตรวจสอบซอฟต์แวร์ เป็นการตรวจสอบความถูกต้องของซอฟต์แวร์ มีความถูกต้องและเป็นไปตามความต้องการ
- การวิวัฒนาการของซอฟต์แวร์เป็น การปรับปรุงและพัฒนาหลังจากที่มีการใช้ซอฟต์แวร์ไปสักระยะหนึ่ง อาจเกิดจากการเปลี่ยนแปลงความต้องการ หรือ เทคโนโลยีที่เปลี่ยนแปลงไป

แบบจำลองการพัฒนาซอฟต์แวร์แบบจำลองน้ำตก

แบบจำลองน้ำตก (Waterfall Model)

จะมีการทำงานแบบทำให้เสร็จสิ้นทีละ Stage แบบ 100 % จากนั้นค่อยเริ่มทำ Stage ถัดไป

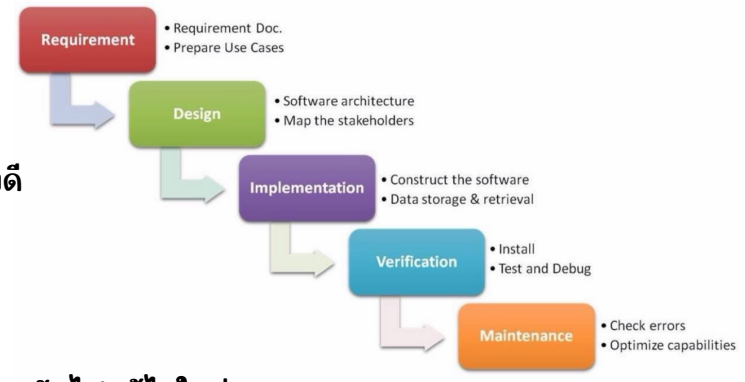
Requirement - จัดเก็บความต้องการของลูกค้า

Design - เป็นการออกแบบกำหนดแผนการทำงาน

Implementation - การพัฒนาระบบงาน (Coding)

Verification - การตรวจสอบ ทดสอบ

Maintenance - การปรับปรุง แก้ไข



ข้อดี

สามารถควบคุมและตรวจสอบแต่ละ Stage ได้เป็นอย่างดี

สามารถกำหนดระยะเวลาของแต่ละ Stage ได้

เหมาะกับงานขนาดเล็กที่ไม่ซับซ้อนมาก

ข้อเสีย

ถ้าผ่าน Stage นั้นไปแล้วแต่มาพบข้อผิดพลาดทำให้ต้องกลับไปแก้ไขใหม่

ในกรณีที่ลูกค้าเปลี่ยน Requirementจะต้องกลับไปแก้ใหม่ตั้งแต่ต้น

ไม่ยืดหยุ่นในกรณีที่มีการเปลี่ยนแปลงอย่างรวดเร็ว

แบบจำลองการพัฒนาซอฟต์แวร์แบบจำลองวิวัฒนาการ

แบบจำลองวิวัฒนาการ (Evolutionary Process Model)

แบบจำลองโปรแกรมต้นแบบ (Prototype Model)

วิศวกรซอฟต์แวร์จัดทำโปรแกรมต้นแบบขึ้นมาเพื่อนำไปทดลองกับผู้ใช้ ลูกค้า หรือผู้ที่เกี่ยวข้อง สำหรับสืบค้นความต้องการที่แน่ชัดว่าถูกต้องตรงกับความต้องการหรือไม่ ต้องเพิ่มเติม แก้ไข ปรับลดหรือไม่ ตรงกับวัตถุประสงค์หรือไม่

ข้อดี

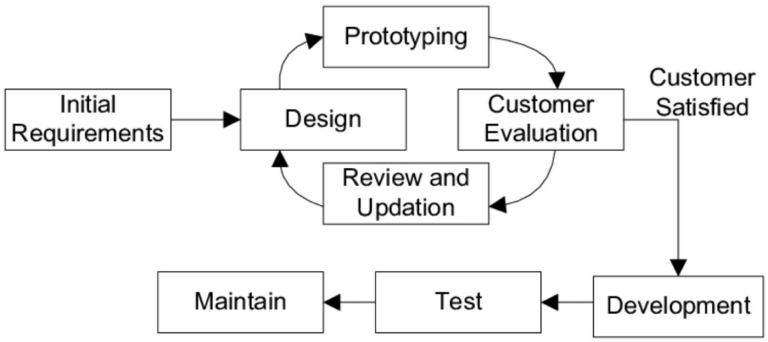
•ช่วยประหยัดเวลา ค่าใช้จ่าย และลด

•ความเสี่ยงที่จะเกิดขึ้นในระหว่างการพัฒนา

ข้อเสีย

• Prototype ซึ่งมีความเสี่ยงในการไม่ถูกยอมรับ จึงอาจทำให้เสียเวลาและไม่ทัน

• มีค่าใช้จ่ายในการทำ Prototype



PROTO TYPE MODEL

แบบจำลองขดหอย (Spiral Model)

มีกระบวนการผลิตซอฟต์แวร์คือ ผลิตในลักษณะวนซ้ำเป็นเกลียววน แต่ละรอบในเกลียววนหมายถึงหนึ่งระยะ (phase) ของกระบวนการซอฟต์แวร์ เมื่อทำไปหลายๆรอบจนกระทั่งซอฟต์แวร์นั้นเสร็จสิ้นสมบูรณ์จึงทำการส่งมอบ รัศมี ของวงกลม จะหมายถึง Cost ที่เกิดขึ้นในขบวนการพัฒนา Software ถ้าจำนวนของ Cycle ที่มากขึ้นก็จะหมายถึง Cost ของการพัฒนาก็จะเพิ่มมากขึ้นด้วย

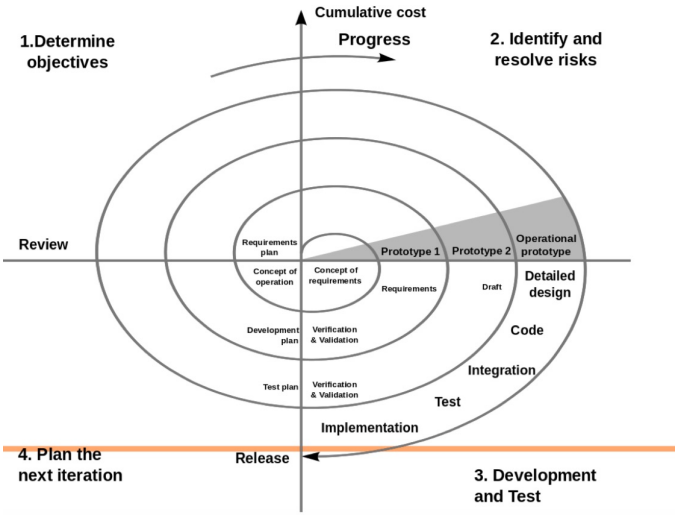
มุม ของวงกลม หมายถึงความก้าวหน้าในการปฏิบัติแต่ละขั้นตอนในแต่ละ Cycle ได้สำเร็จส่วนต่าง ๆ ในแต่ละ Cycle ของ Spiral Model ประกอบด้วย

1.Determine Phase : การวิเคราะห์ความต้องการ

2.Evaluate Phase / Resolve Risk : การวิเคราะห์ความเสี่ยง

3.Develop, verify Phase: การพัฒนาและตรวจสอบ

4.Next Phase: การวางแผนในกระบวนการถัดไป



ข้อดี

วิเคราะห์ความเสี่ยง และการที่ทีมงานต้องติดต่อประสานงานกับลูกค้าอยู่สม่ำเสมอ ช่วยให้รู้ทันความต้องการลูกค้าได้อย่าง ซึ่งแบบจำลองชนิดนี้เหมาะกับการพัฒนาซอฟต์แวร์ที่มีความซับซ้อนสูง

ข้อเสีย

เป็นแบบจำลองที่มีค่าใช้จ่ายสูง เนื่องจากมีความไม่แน่นอนในวงรอบการทำซ้ำ

การวิเคราะห์ความเสี่ยงจำเป็นต้องมีความเชี่ยวชาญเฉพาะด้าน สูงความสำเร็จของโครงการขึ้นอยู่กับขั้นตอนการวิเคราะห์ความเสี่ยงไม่ทำงานได้ดีสำหรับโครงการขนาดเล็ก

แบบจำลองทำเพิ่ม (Incremental Model)

มีหลักการคือการแบ่งระบบงานออกเป็นระบบย่อยต่าง ๆ โดยระบบย่อยเรียกว่า Incrementโดยจะทำการพัฒนาระบบงานที่เป็นงานหลักของระบบและต่อเติมแต่ละ Incrementจนกระทั่งได้ระบบงานที่เสร็จสมบูรณ์

ขั้นตอนการทำงานของ Incremental Model

- การศึกษาความเป็นไปได้ของระบบ
- การวางแผนและการกำหนดความต้องการ
- ขั้นตอนการออกแบบระบบ (Product Design โดยพัฒนาและตรวจสอบระบบย่อยทีละระบบ

การพัฒนาระบบย่อยประกอบด้วยขั้นตอนการทำ งาน 5 ขั้นตอน และมีทวนซ้ำ

ขั้นตอนการทำงานของแต่ละรอบประกอบด้วย

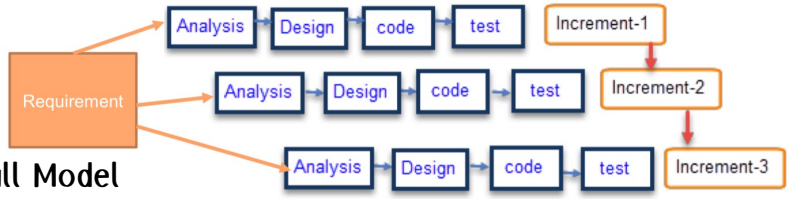
1.การออกแบบรายละเอียดของระบบย่อย พร้อมทั้งตรวจสอบความถูกต้อง

2.เขียนโปรแกรมและทดสอบโปรแกรมหน่วยย่อยต่างๆ (Unit Testing)

3.นำโปรแกรมย่อยต่างๆ มาประกอบรวมกันและตรวจสอบว่าทำงานได้อย่างถูกต้องหรือไม่

4.การนำระบบไปใช้งานจะมีการทดสอบระบบว่าระบบ ทำงานได้อย่างถูกต้องและเป็นไปตามความต้องการของผู้ใช้

5.ขั้นตอนการดำเนินงานและบำรุงรักษา จะเป็นการทบทวนเพื่อตรวจสอบ ความถูกต้อง ว่าระบบตรงตามความต้องการของผู้ใช้หรือไม่ (Revalidation)



ข้อดี

สามารถส่งมอบงานได้เร็ว

มีความยืดหยุ่นในการพัฒนาระบบงานมากกว่า Waterfall Model

เห็นชิ้นงานที่มีความคืบหน้าในทุกๆ Increment จะได้ส่วนย่อยของระบบซอฟต์แวร์ขึ้นมา

ข้อเสีย

หากมีการออกแบบหรือวางแผนในแต่ละ increment ไม่ดี อาจจะทำให้ส่วนย่อยๆ ไม่สามารถทำงานร่วมกันได้

แบบจำลองการพัฒนาซอฟต์แวร์แบบจำลองการพัฒนาแอปพลิเคชันแบบเร่งรัด

แบบจำลองชนิดนี้มีชื่อย่อว่า RADคือ แยกชิ้นงานออกเป็นส่วนๆหรือโมดูล แล้วแบ่งโมดูลเหล่านี้ให้กับทีมงานหลายๆ ทีมพัฒนาทำงานคู่ขนานกันไปเพื่อให้แล้วเสร็จภายในระยะเวลาอันรวดเร็ว เมื่อทำเสร็จก็จะนำในส่วนเหล่านี้มาประกอบเป็นชิ้นงานใหม่นั้นอีกครั้งวิศวกรซอฟต์แวร์จะต้องมีความรู้และทักษะในการใช้เทคโนโลยีคอมโพเน้นท์ การนำกลับมาใช้ใหม่ การทำงานแบบมีส่วนร่วม และการวางแผนโครงการที่ดี

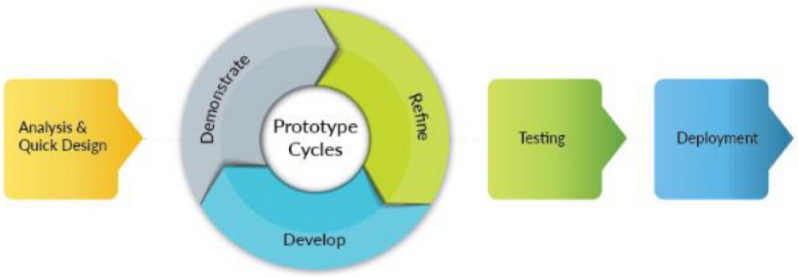
ขั้นตอนการทำงานประกอบไปด้วย 4 ส่วน ดังนี้

1.การกำหนดความต้องการ เป็นการกำหนดหน้าที่ภายในระบบ

2.การออกแบบโดยผู้ใช้

3.การสร้างระบบโดยการใช้ตัวชอฟต์แวร์ประยุกต์อย่างเร็ว

4.การเปลี่ยนระบบ ทำการทดสอบระบบให้เสร็จสิ้นก่อนฝึกอบรมแล้วจึงมีการเปลี่ยนแปลงเครื่องมือในการพัฒนาระบบงานอย่างเร็ว



แบบจำลองการพัฒนาซอฟต์แวร์บบจำลองกระบวนการแบบผสมผสาน

แบบจำลองกระบวนการแบบผสมผสาน (Rational Unified Process Model : RUP model

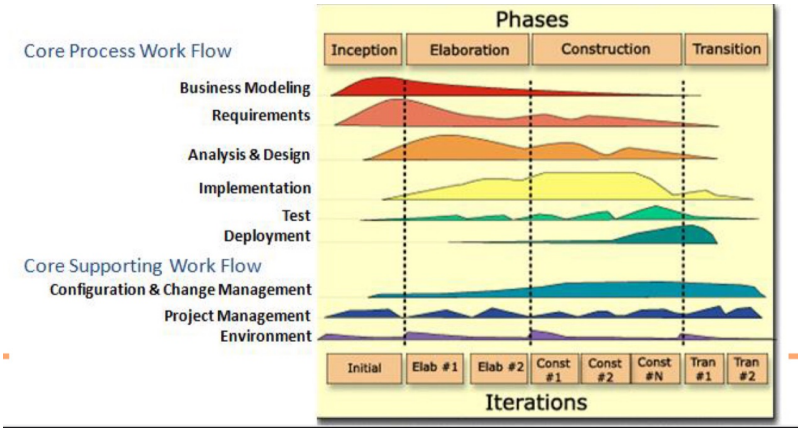
- การใช้งาน UML (Unified Modeling Language)
- การพัฒนาแบบวนซ้ำ (Iterative Development)
- การจัดการความเสี่ยง (Risk Management)
- การจัดการการเปลี่ยนแปลง (Change Management)

RUP แบ่งวงจรชีวิตการพัฒนาซอฟต์แวร์ออกเป็น 4 เฟสหลัก:

1 Inception: ระยะเริ่มต้น มุ่งเน้นไปที่การกำหนดวิสัยทัศน์ เป้าหมาย และขอบเขตของโครงการ
2 Elaboration: ระยะขยายความ มุ่งเน้นไปที่การวิเคราะห์ความต้องการ การออกแบบระบบ และสร้างต้นแบบ
3 Construction: ระยะสร้าง มุ่งเน้นไปที่การพัฒนาซอฟต์แวร์ การทดสอบ และการแก้ไขข้อผิดพลาด
4 Transition: ระยะเปลี่ยนผ่าน มุ่งเน้นไปที่การติดตั้ง การใช้งาน และการบำรุงรักษาซอฟต์แวร์

แต่ละเฟสประกอบด้วย

- Workflows: กิจกรรมต่างๆ ที่ต้องดำเนินการ
- Artifacts: ผลลัพธ์ที่คาดหวัง
- Milestones: จุดตรวจสอบความคืบหน้า



ข้อดีของ RUP:

- มีความยืดหยุ่น ปรับเปลี่ยนได้ตามสถานการณ์
- รองรับการเปลี่ยนแปลงความต้องการของผู้ใช้
- ควบคุมความเสี่ยงและข้อผิดพลาดได้ดี
- เหมาะกับโครงการขนาดใหญ่และซับซ้อน

ข้อเสียของ RUP:

- ออกแบบและจัดการโครงการยาก
- ควบคุมความคืบหน้าและงบประมาณได้ยาก

แบบจำลองการพัฒนาซอฟต์แวร์แบบจำลองเชิงชิ้นส่วน

แบบจำลองเชิงชิ้นส่วน (Component-Based Development Process)

แบบจำลองที่อาศัยคอมโพเน้นท์ (Component-Based Software Engineering: CBSE) เป็นแนวทางการผลิตซอฟต์แวร์ที่อาศัยคอมโพเน้นท์ที่ได้ถูกสร้างไว้แล้วมาประกอบเป็นซอฟต์แวร์ใหม่ที่ต้องการด้วยหลักการนำกลับมาใช้ใหม่ (Reuse) ทั้งในส่วนของโค้ดและสถาปัตยกรรม

ข้อดี

ลดต้นทุนในการผลิตลงได้มาก และมั่นใจได้ว่าซอฟต์แวร์มีคุณภาพแน่นอน

ข้อจำกัด

วิศวกรซอฟต์แวร์จะต้องมีความรู้และทักษะในการใช้เทคโนโลยีคอมโพเน้นท์ การนำกลับมาใช้ใหม่

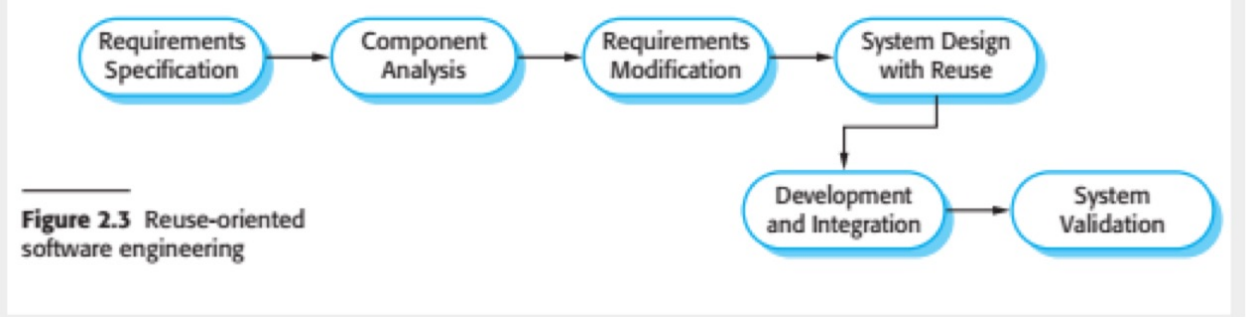


Figure 2.3 Reuse-oriented software engineering

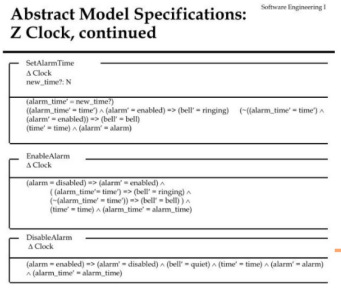
แบบจำลองการพัฒนาซอฟต์แวร์แบบวิธีการ

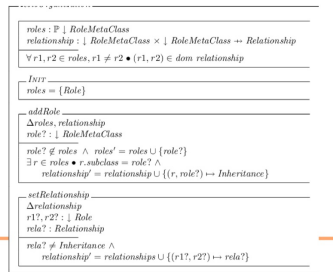
แบบจำลองแบบวิธีทางการ (Formal Method Model)

เป็นการรวมเอาชุดกิจกรรมข้อกำหนดทางคณิตศาสตร์แบบเป็นทางการของซอฟต์แวร์ ทำให้ซอฟต์แวร์สามารถกำหนดพัฒนา และตรวจทานระบบคอมพิวเตอร์ได้โดยใช้สัญลักษณ์ทางคณิตศาสตร์ที่เข้มงวดเหมาะกับซอฟต์แวร์ที่ต้องการความปลอดภัยสูง เช่น ซอฟต์แวร์การแพทย์ หรือการควบคุมการบิน

ข้อดี

ลดความคลุมเครือ ความไม่สมบูรณ์และความไม่คงเส้นคงวา ข้อผิดพลาดจะถูกค้นพบและแก้ไขได้อย่างเป็นระบบ





ข้อด้อย

การพัฒนาแบบจำลองนี้ในปัจจุบันใช้เวลาและค่าใช้จ่ายมาก นักพัฒนาซอฟต์แวร์ทั่วไปไม่มีความรู้พื้นฐานที่จะใช้งานวิธีนี้ จึงต้องเสียค่าใช้จ่ายในการอบรมมาก

มีความลำบากในการใช้แบบจำลองนี้เป็นกลไกในการสื่อสารระหว่างผู้พัฒนากับลูกค้า

แบบจำลองการพัฒนาซอฟต์แวร์แบบจำลองอไจล์

แบบจำลองอไจล์ (Agile Model)

3 ตำแหน่งสำคัญ ในแบบจำลองเอไจน์ คือ

Product Owner : มีหน้าที่ประเมิน Values และจัด Priorities ของ Tasks ต่างๆ ให้กับทีม

Scrum Master : เป็นผู้ทำให้การทำงานเป็นไปอย่างลื่นไหล กำจัดอุปสรรคที่ขัดขวางไม่ให้ทีมบรรลุเป้าหมาย

Team : จะทำงานแบบ Self-Management ซึ่งในหนึ่งทีมจะประกอบด้วยคนประมาณ 3-9คน และรวมทุกตำแหน่งทั้ง Designer, Programmer, UV/UX, Testing เข้าด้วยกัน เพื่อให้ทีมหนึ่งทีมสามารถทำงานตั้งแต่ต้นจนจบได้ด้วยตัวเอง โดยไม่ต้องข้ามแผนก

วิธีการทำงานของ Scrum Framework ประกอบด้วย

Backlog : เป็นแผนงานที่ต้องทำ ทั้งความต้องของลูกค้าและทีม ซึ่ง Product Owner จะเป็นคนตัดสินใจ

Sprint Phase: ส่งงานให้เร็วและบ่อย ซึ่ง Period นั้นจะเรียกว่า Sprint โดยมีกำหนดประมาณ 2-4 สัปดาห์ ซึ่งเมื่อจบ Sprint ก็จะมีการ Review ความก้าวหน้าให้ลูกค้าทราบ

Scrum Meeting : ในทุกๆ เช้าทีมจะมีการประชุมสั้นๆ 10-15 นาที เพื่อบอกว่าเมื่อวานทำอะไร วันนี้จะทำอะไร และมีปัญหาอะไรบ้าง รู้ว่ากำลังเดินเข้าสู่เป้าหมายหรือยัง และมีการแก้ไขปัญหาอย่างต่อเนื่อง

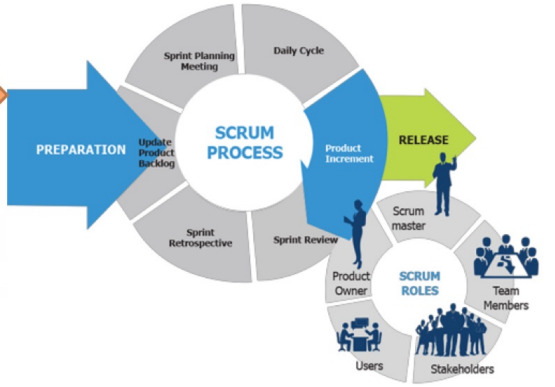
ข้อดี

เน้นการทำงานเป็นทีม การมีส่วนร่วม

เน้นการพัฒนางานที่มีคุณภาพตรงความต้องการของลูกค้าและรวดเร็ว

ข้อจำกัด

ต้องมีการจัดทีมงานที่สนับสนุนการทำงานแบบอไจล์ การสร้างความร่วมมือระหว่างทีม



เครื่องมือในกระบวนการพัฒนาซอฟต์แวร์

ซอฟต์แวร์ ซึ่ง CASE (Computer Aided Software Engineering) Tools เป็นซอฟต์แวร์หรือเครื่องมือที่เป็นเครื่องมือที่มีส่วนประกอบช่วยสนับสนุนการทำงานในกิจกรรมต่างๆของงานวิศวกรรมซอฟต์แวร์ สามารถช่วยแบ่งเบาภาระของนักพัฒนาระบบลงได้ ช่วยให้ทีมงานสามารถทำงานซ้ำๆ เดิมได้ง่ายและรวดเร็ว

CASE Tools ออกเป็น 8 กลุ่ม ดังนี้

1) เครื่องมือสำหรับงานความต้องการซอฟต์แวร์ แบ่งออกเป็น 2 กลุ่มย่อยคือ

เครื่องมือในการสร้างแบบจำลองความต้องการ เป็นเครื่องมือที่ วิเคราะห์ กำหนด และตรวจสอบความต้องการด้านซอฟต์แวร์เป็นต้น

เครื่องมือการติดตามความต้องการ เป็นเครื่องมือที่ใช้ติดตามความต้องการที่เปลี่ยนแปลงอยู่ตลอดเวลา

2) เครื่องมือสำหรับงานออกแบบซอฟต์แวร์ เป็นซอฟต์แวร์ที่ใช้ สร้างและตรวจสอบงานออกแบบซอฟต์แวร์มีหน้าที่สนับสนุนการวิเคราะห์ความต้องการด้านซอฟต์แวร์ ยกตัวอย่างเช่น Rational Rose, EA เป็นต้น

3) เครื่องมือสำหรับงานสร้างซอฟต์แวร์ เป็นกลุ่มซอฟต์แวร์ที่สนับสนุนงานในการสร้างซอฟต์แวร์ทั้งหมด ได้แก่ เครื่องมือแก้ไขโปรแกรมคอมไพเลอร์ อินเตอร์พรีเตอร์ ดีบักเกอร์ ไม่ว่าจะเป็นการสร้างซอฟต์แวร์แบบบนเครื่องเดียวแบบไคลเอนท์เซิร์ฟเวอร์ยกตัวอย่างเช่น Edit Plus, Eclipse, Dream Weaver, Windev, Visual Studio เป็นต้น

4) เครื่องมือสำหรับทดสอบซอฟต์แวร์ ได้แก่

- เครื่องสร้างกรณีทดสอบ ใช้สร้างกรณีทดสอบซอฟต์แวร์
- กรอบการปฏิบัติการทดสอบ ใช้ทดสอบซอฟต์แวร์ภายใต้สภาพแวดล้อมล่วงหน้า
- เครื่องมือประเมินผลการทดสอบ ใช้สนับสนุนการประเมินผลการทดสอบ
- เครื่องมือบริหารงานทดสอบ เป็นเครื่องมือสนับสนุนทุกกิจกรรม
- เครื่องมือวิเคราะห์ประสิทธิภาพการทดสอบ เป็นเครื่องมือที่ใช้วัดผลและวิเคราะห์ประสิทธิภาพการทำงาน

5) เครื่องมือบำรุงรักษาซอฟต์แวร์ ต้องทำการซ่อมบำรุงรักษาซอฟต์แวร์ ให้คงสภาพที่ใช้การได้อย่างดี ซึ่งงานซ่อมบำรุงนี้ งานบางส่วนจะทำเหมือนงานสร้างซอฟต์แวร์แต่มีการทำงานส่วนบำรุงรักษาเพิ่มเติม

แบ่งเป็น 2 กลุ่มได้แก่

- เครื่องมือสร้างความเข้าใจ ให้ทีมซ่อมบำรุงเข้าใจโปรแกรมของซอฟต์แวร์ได้ง่ายขึ้น นั่นคือมีเครื่องมือส่วนที่จัดการกับโค้ดของโปรแกรม อันได้แก่ เครื่องมือแก้ไข โปรแกรม คอมไพเลอร์ อินเตอร์พรีเตอร์ ดีบักเกอร์

-เครื่องมือรื้อปรับระบบใหม่ เป็นเครื่องมือที่ช่วยในการรื้อโครงสร้างของซอฟต์แวร์ทีละส่วน เพื่อนำมาปรับหรือแก้ไข

6) เครื่องมือจัดการโครงแบบ เป็นเครื่องมือที่ใช้ติดตามการเปลี่ยนแปลงของทุกองค์ประกอบของซอฟต์แวร์ จัดการรุ่นของซอฟต์แวร์และวางจำหน่วยซอฟต์แวร์

7) เครื่องมือสำหรับบริหารงานวิศวกรรมซอฟต์แวร์ ได้แก่

- เครื่องมือวางแผนและติดตามโครงการได้แก่ซอฟต์แวร์ที่ใช้ในการประมาณการแรงงาน และต้นทุน
- เครื่องมือจัดการความเสี่ยง ได้แก่ ซอฟต์แวร์ที่ระบุปัจจัยเสี่ยงประมาณการผลกระทบและติดตามความเสี่ยง
- เครื่องมือวัดผลโครงการได้แก่ ซอฟต์แวร์ที่ใช้ในการวัดผลทุกกิจกรรมของโครงการ

8) เครื่องมือสำหรับคุณภาพซอฟต์แวร์ แบ่งเป็น 2 กลุ่มได้แก่

- เครื่องมือตรวจสอบคุณภาพ ได้แก่ เครื่องมือที่ใช้ทบทวนและตรวจสอบคุณภาพของซอฟต์แวร์
- เครื่องมือวิเคราะห์คุณภาพ ได้แก่เครื่องมือที่ใช้วิเคราะห์ลักษณะด้านต่างๆ ของซอฟต์แวร์

การทดสอบซอฟต์แวร์ (Software Testing)

เป็นการทดสอบความสมบูรณ์ของโปรแกรม รวมทั้งความน่าเชื่อถือและความถูกต้องของผลลัพธ์จากโปรแกรมที่พัฒนาขึ้นตรวจสอบประสิทธิภาพการทำงานของซอฟต์แวร์เพื่อประเมินและปรับปรุงคุณภาพของซอฟต์แวร์ โดยการหาข้อผิดพลาดและปัญหาที่เกิดขึ้นและทำการแก้ไขปัญหา

Testing & Debug

Testing คือการหาข้อผิดพลาดว่ามีหรือไม่

Debug คือการหาตำแหน่งของข้อผิดพลาด (เพื่อจะได้แก้ไข)

Software Testing

Error หรือ Mistake : เกิดจากคน เช่น ใช้คำสั่งผิด พิมพ์คำสั่งผิด กรณีนี้จะเรียกว่า bug

Fault หรือ Defect : เป็นผลลัพธ์ที่เกิดขึ้นจาก error หรือเรียกว่า " fault "

Failure : จะเกิดขึ้นเมื่อมีการประมวลผล Fault ทำให้ระบบล้มเหลว

Incident : อาการผิดปกติที่บ่งบอกให้รู้ว่ามี failure เกิดขึ้น

Test case : หรือที่เรียกว่ากรณีทดสอบ จะต้องมีลักษณะที่สัมพันธ์กับพฤติกรรมของโปรแกรมนั้นๆ หลักๆแล้วจะประกอบด้วย ข้อมูลที่ input เข้าไปในโปรแกรม และมีผลลัพธ์ที่คาดว่าจะได้จากการป้อน input เข้าไป

ข้อผิดพลาด(error)

- Syntax error ที่เกิดจากการเขียนโปรแกรมผิด Syntax ของ ภาษา เกิดขึ้นตอน Compile โปรแกรม
- Logical error เกิดจากการเขียนโปรแกรมเขียนผิดตรรกะ เกิดตอนโปรแกรมปฏิบัติงาน (execute)
- Runtime error เกิดขึ้นขณะที่โปรแกรมปฏิบัติงานไม่มี Syntax error แต่มี Logical error อยู่

กลวิธีการทดสอบซอฟต์แวร์

- กลยุทธ์การทดสอบซอฟต์แวร์ – วิธีการออกแบบกรณีทดสอบ (Test case) และการวางแผนทดสอบ (Test Plan) เพื่อให้ได้ชุด ของขั้นตอนตามที่ปฏิบัติ เป็นการยืนยันว่าการสร้างซอฟต์แวร์ ประสบผลสำเร็จ
- กลยุทธ์ใดๆ ต้องมีแผนการทดสอบ การออกแบบกรณีทดสอบ การลงมือทดสอบ และการรวบรวมและประเมินผลข้อมูลผลลัพธ์

หลักการและเป้าหมายของการทดสอบ คือ การหาความผิดพลาดให้พบ กรณี ทดสอบที่ดีควรมีความเป็นได้สูงที่จะหาข้อผิดพลาดพบ

Test case

- บอกถึงสถานการณ์ต่างๆ ที่โปรแกรมต้องตอบสนอง
- ครอบคลุมตั้งแต่ สถานการณ์เริ่มต้น สถานการณ์ต่างๆ ที่มีผลต่อ การดำเนินการของโปรแกรมและผลลัพธ์ที่คาดหวัง

เทคนิคการทดสอบโปรแกรม

Manual Testing การทดสอบโดยไม่ใช้เครื่องคอมพิวเตอร์

แบ่งได้เป็น 2 ชนิดคือ

1) Inspection

- โปรแกรมเมอร์ตรวจสอบเอง
-เปรียบเทียบ Code ของโปรแกรมที่เขียนขึ้นกับ รายการอื่น
- เป็นการทดสอบความผิดพลาดของ Code เท่านั้น
- ป้องกันข้อผิดพลาดรูปแบบเดิมไม่ให้เกิดซ้ำอีก
- ไม่ทำให้ทราบว่ามีผลลัพธ์ถูกต้องหรือไม่

2) Desk Checking

- การตรวจสอบโค้ดของโปรแกรมตามลำดับคำสั่งว่ามีตรรกะ ผิดพลาดหรือไม่
- ทำโดยผู้ที่ได้รับการแต่งตั้งให้เป็นผู้ทดสอบ
- วิธีการนี้ไม่เหมาะสำหรับงานที่ความซับซ้อนสูง เนื่องจากเสียเวลา มากในการทดสอบ

Automated Testing

Syntax Checking ตรวจสอบไวยากรณ์ที่เขียนขึ้นโดยใช้ Compiler ใช้เวลาไม่นาน ไม่สามารถตรวจสอบผลลัพธ์ได้
Unit Testing เรียกอีกชื่อว่า Module Testing โปรแกรมเมอร์ทดสอบ Module หาข้อผิดพลาดในการทำงานแต่ละโมดูลมีใช้ 2 แบบคือ

Black Box Testing

White Box Testing

Black Box Testing

· เป็นการทดสอบโดยไม่คำนึงถึงคำสั่ง เป็นการทดสอบ Function ต่างๆ ของโปรแกรมตาม Requirements

· เป็นการทดสอบโดยดูค่า Output จาก Input ที่ให้กับโปรแกรมต้องมีความสอดคล้องกัน

การกำหนดข้อมูลในการทดสอบ ได้แก่

- ค่าตัวแทนของกลุ่ม
- ค่าสูงสุด
- ค่าต่ำสุด
- ค่าเกินพิกัด
- ค่าที่ผิดวิสัย

สำหรับ Technique ต่างๆ ที่ใช้ในการทดสอบแบบ Black box testingนั้นมีหลายวิธี แต่จะขอตัวอย่างที่เด่นๆ
Equivalence partitioning เป็นการกำหนดค่าตัวแทน ของกลุ่มข้อมูลขึ้นมา 1 ค่า แล้วนำค่านั้น มาใช้ในการทดสอบ ซึ่งจะสามารถ Apply ได้ว่าถ้าใช้ค่าตัวแทนนี้มาทำการทดสอบได้ผลลัพธ์อย่างไร ค่าอื่นๆ ที่อยู่ภายใต้กลุ่มนี้ ก็จะมีผลลัพธ์เช่นเดียวกัน
Boundary value analysis เป็นวิธีการทดสอบโดยกำหนดขอบเขตของข้อมูลขึ้นมา เพื่อจะได้ค่าinput data ที่ครอบคลุม เช่น ระบบธนาคารสามารถให้โอนเงินผ่าน ATM ขั้นต่ำคือ 100 และสูงสุดคือ 500 บาท การทดสอบจะต้องกำหนดขอบเขตของข้อมลที่จะต้องนำมาทดสอบ

White Box Testing

· เป็นการทดสอบเพื่อดูโครงสร้างของโปรแกรม หรือทางเดินในโปรแกรม

· ต้องสร้างชุดทดสอบเฉพาะสำหรับทดสอบในเงื่อนไขต่างๆ โดยชุดทดสอบจะต้องประกอบด้วยชุดที่สามารถประมวลผลอย่างปรกติและไม่ปรกติ

· ต้องมีความรู้เรื่องของระบบว่ามีการออกแบบอย่างไร ทำงานอย่างไร

· ต้องมีความรู้ในเรื่องของ Programming

Basis Path Testing เป็น idea ที่ช่วยในการทำ White-box ให้ง่ายขึ้น เปลี่ยน Flowchart ให้เป็น Graph ที่ประกอบด้วย Vertices และEdgeเพื่อทดสอบความสลับซับซ้อนทางตรรกะวิทยาโดยจะทดสอบทุกๆ "Execution path" การออกแบบกรณีทดสอบจะทำการทดสอบ"Basic set" ที่ใช้
"Execute logical path" ต่างๆ คือ

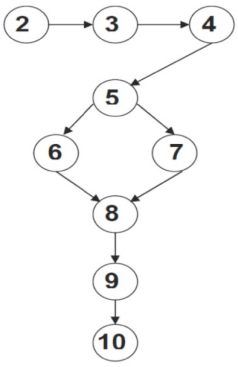
1) Sequence logical path
2) If logical path
3) Loop logical path

การสร้างกรณีทดสอบ (test case)

สร้างกรณีทดสอบ (test case) ที่จะนำมาทดสอบในแต่ละ path เลือกข้อมูลให้มีค่าเหมาะสมในการทดสอบแต่ละเส้นทางเมื่อกำหนดกรณีทดสอบให้กับทุก path แล้วให้ทำการทดสอบแต่ละ path และเปรียบเทียบผลลัพธ์ที่ได้ กับ ผลลัพธ์ที่คาดหวัง(Expected results)

Path Testing

1. *Program 'Simple Subtraction'*
2. *Input (x, y)*
3. *Output (x)*
4. *Output (y)*
5. *If x > y then DO*
6. *x – y = z*
7. *Else y – x = z*
8. *Endlf*
9. *Output (z)*
10. *Output "End Program"*



Automated Testing

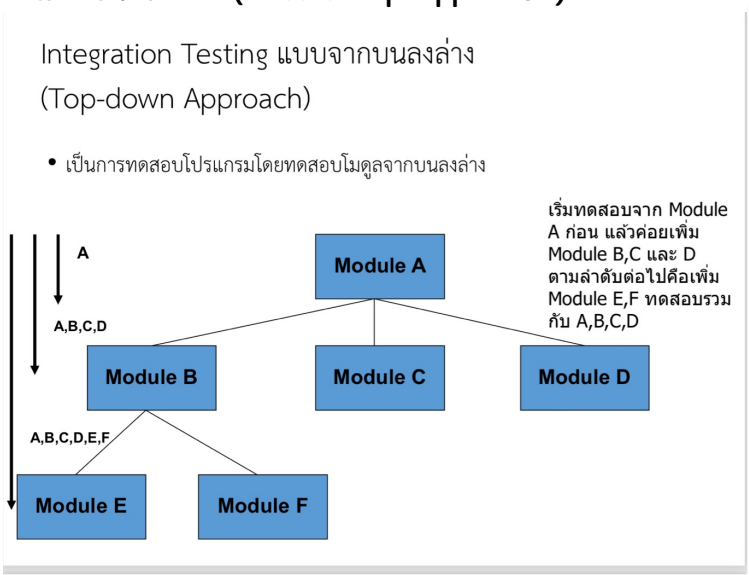
Integration Testing

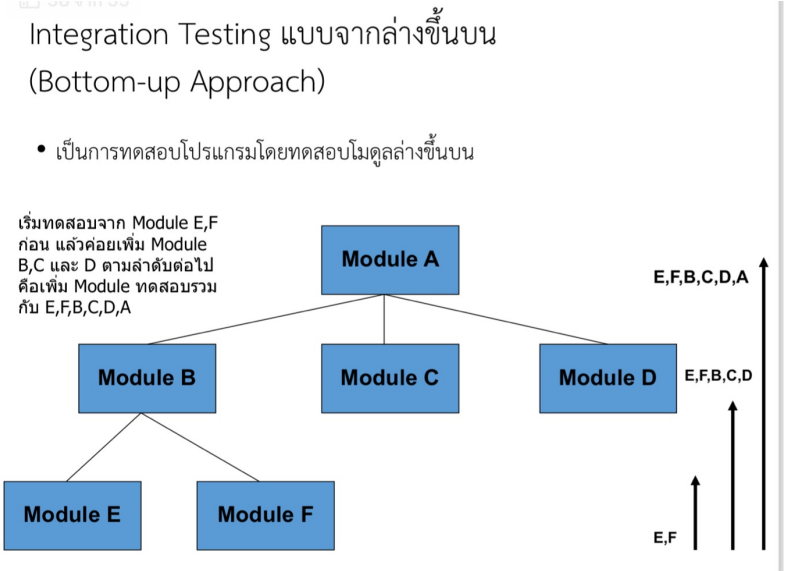
เพื่อตรวจสอบความถูกต้องของ Function การทำงานต่างๆ เมื่อมีการIntegrate unit / module เข้าร่วมกันทดสอบ

การทำงานรับส่งข้อมูลกันระหว่างโมดูลอาศัย structure chart มี 2 แบบ คือ

แบบบนลงล่าง (Top-down Approach)

แบบล่างขึ้นบน (Bottom-up Approach)





Stub Testing

การทดสอบแบบเพิ่มโมดูล (Integration Testing) นั้นจะค่อย ๆ เพิ่มโมดูลเข้ามาทดสอบตามลำดับ กรณีที่เพิ่มจากบนลงล่าง คือ เริ่มทดสอบจาก Module A ก่อน แล้วค่อยเพิ่ม Module B,C และ D ตามลำดับก่อนเพิ่ม Module E,F แต่ความจริง Module B ต้องเรียกใช้ข้อมูลจาก

Module E,F ที่อยู่ในระดับล่างซึ่งยังไม่ถึงรอบการทดสอบ

ปัญหาคือ อาจไม่เห็นผลลัพธ์ ดังนั้นจึงต้องสร้าง Stub Testing เป็นตัวแทน Module E,F เพื่อทดสอบชั่วคราว

Stub คือ กลุ่มคำสั่งสั้นๆ ที่เขียนขึ้นมาเพื่อเป็นโมดูลตัวแทนในการทดสอบโปรแกรม

System Testing

คล้ายการทดสอบแบบ Integration แต่จะต่างกันตรงที่จะทดสอบโปรแกรมหนึ่ง และเพิ่มเรื่อยๆจะครบทุกโปรแกรมของระบบงานเป็นการตรวจสอบการทำงานทั้งระบบ ว่าโปรแกรมทุกโปรแกรมเมื่อทำงานร่วมกันแล้วจะให้ผลลัพธ์ที่ถูกต้องแล้วหรือไม่ จุดประสงค์ เพื่อตรวจสอบระบบว่าระบบทำงนได้ถูกต้องและได้ผลลัพธ์ตรงตาม Requirement

กลยุทธ์ในการทดสอบประสิทธิภาพของระบบ

1. การทดสอบการทำงานสูงสุด (Peak Load Testing)

เป็นการทดสอบประสิทธิภาพในการประมวลผลของระบบ เมื่อมีการทำรายการมากที่สุด ณ ช่วงเวลาใดเวลาหนึ่ง

2.การทดสอบประสิทธิภาพของเวลา (PerformanceTesting)

เป็นการทดสอบระบบเพื่อพิจารณาถึงช่วงเวลาที่ใช้ในการประมวลผลรายการ ว่าใช้ระยะเวลานานเพียงใดในการทำรายการ

3. การทดสอบการกู้ระบบ (Recovery Testing)

เป็นการทดสอบความสามารถในการกู้ระบบกรณีที่ระบบล้ม รวมถึงความสามารถในการกู้ข้อมูลด้วย

4) การทดสอบการเก็บข้อมูล (Storage Testing)

เป็นการทดสอบว่า เก็บข้อมูลได้สูงสุดเป็นจำนวนเท่าใด

5) การทดสอบกระบวนการ (Procedure Testing)

เป็นการทดสอบการจัดทำคู่มือการดำเนินงานของระบบ และคู่มือการใช้งานว่าสามารถสร้างความเข้าใจให้กับผู้ใช้หรือไม่ และสามารถใช้เมื่อเกิดปัญหาหรือไม่

6) การทดสอบผู้ใช้ (User Testing)

เป็นการทดสอบการใช้งานจริงของระบบเพื่อต้องการทราบว่าปัญหาในการใช้ระบบเป็นอย่างไรบ้าง

การทดสอบการกู้คืน (Recovery Testing)

ความสามารถในการกู้คืนเมื่อระบบเกิดความล้มเหลว Fault Tolerance ระบบต้องทำงานต่อได้และต้องทนต่อความผิดพลาดการทดสอบ ทำให้ระบบล้มเหลวในสถานการณ์ต่างๆแล้วดูการกู้คืนของข้อมูล

การทดสอบการรักษาความปลอดภัย(Security Testing)

มีเครื่องมือในการรักษาความปลอดภัยของระบบเป็นที่น่าพอใจหรือไม่ การรักษาความปลอดภัยอย่างง่ายได้แก่ การใช้วิธีการให้สิทธิและ การควบคุมสิทธิ การเข้ารหัสข้อมูล

การทดสอบแรงตึงเครียด (Stress Testing)

เป็นการทดสอบในสถานการณ์ที่ไม่เป็นปรกติ เพื่อดูความทนทานในระบบ แรงตรึงเครียดของระบบอาจจะเกิดจากการนำเข้าข้อมูลที่มากเกินไปการประมวลผลที่บ่อยเกินไปSensitivity Testing

การทดสอบสมรรถนะ (Performance Testing)

เป็นการวัดสมรรถนะของระบบว่าอยู่ในระดับที่ยอมรับได้เช่น ระยะเวลาในการตอบสนองของงาน การจัดสรพื้นที่หน่วยความจำการสร้าง Log File เพื่อเป็นการจดบันทึกเพื่อวัดสมรรถนะการ ทำงานของระบบในแต่ละช่วงเวลา

วิธีการประเมินผลการทำงานของระบบ

·การใช้แบบสอบถาม

· การบันทึกเทปการทำงานของผู้ใช้

· การสร้างส่วนพิเศษภายในระบบ ให้สามารถบันทึกข้อมูลเกี่ยวกับการทำงานของผู้ใช้

· การสร้างระบบให้ผู้ใช้สามารถบันทึกความคิดเห็นของตนขณะกำลังใช้งานระบบนั้น ๆ

การทดสอบการยอมรับของผู้ใช้

· หลังจากที่ได้ทดสอบความสมบูรณ์และความถูกต้องของโปรแกรม เรียบร้อยแล้ว
. มีขั้นตอนที่มีความสำคัญไม่น้อยไปกว่าการทดสอบโปรแกรม นั้นก็คือการทดสอบการยอมรับจากผู้ใช้
· เนื่องจากการพัฒนาระบบนั้น ก็เพื่อต้องการตอบสนองความต้องการในการดำเนินงานของผู้ใช้ระบบ
· โดยวิธีการทดสอบการยอมรับของระบบนั้นสามารถแบ่งออกเป็น 2ประเภทคือการทดสอบการยอมรับของผู้ใช้
1) Alpha Testingการทดสอบความสมบูรณ์ของระบบโดยผู้ใช้พร้อมกับจะมี Tester/ QAเป็นคนแนะนำ
- ผู้ใช้จะทดสอบระบบขณะยังไม่ได้ติดตั้งในสถานที่จริง โดยทดสอบภายใน
สถานการณ์จำลองที่กำหนดขึ้นโดยทีมงานพัฒนาระบบ ทีมงานจะบันทึก
ข้อผิดพลาดและทำการแก้ไข
- ทำให้ได้รับรู้ว่าระบบมีข้อผิดพลาดอะไรเกิดขึ้นบ้าง
มีการทดสอบ 4 ประการคือ
1. Recovery Testing เป็นการทดสอบการกู้ระบบ
2. Security Testing เป็นการทดสอบความปลอดภัยของระบบ
1. Stress Testing เป็นการทดสอบประสิทธิภาพการทำงานของระบบภายใต้ความกดดัน
4.Performance Testing เป็นการทดสอบประสิทธิภาพการทำงานของระบบภายใต้สภาพแวดล้อมของคอมพิวเตอร์
2) Beta Testing
- การทดสอบความสมบูรณ์ของระบบโดยผู้ใช้ และใช้ข้อมูลจริงในการทดสอบและภายใต้สถานการณ์ที่เกิดขึ้นจริง
โดยไม่มีทีมงานไป เฝ้าสังเกต
- ต้องมีคนคอยจดบันทึกข้อผิดพลาดให้กับทีมงาน
- การทดสอบประเภทนี้ถือว่าเป็นการซ้อมติดตั้งระบบเพื่อใช้งานจริง
- เนื่องจากเป็นการทดสอบระบบอย่างสมจริงไม่ว่าจะเป็น สถานการณ์ข้อมูล ขั้นตอนการดำเนินงาน เอกสารคู่มือ
การฝึกอบรม การสนับสนุนการทำงาน รวมทั้งเป็นการแก้ปัญหาที่เกิดขึ้นระหว่าง Alpha Testing อีกด้วย